Paradigms for M&S Development

RPG Special Topic

11/30/00

Table of Contents

This document corresponds to the web version of the VV&A RPG Special Topic of the same name and date. It has been modified for printing.

Introduction

At the very beginning of a model or simulation development process, the Developer and M&S Program Manager (PM) should identify a development paradigm (i.e., a pattern or template used to describe a development process) to follow during planning and development of the simulation. This paradigm should address and accommodate as closely as possible the unique set of circumstances associated with the M&S program, such as the schedule, resources available, overall requirements and objectives, and level of risk or uncertainty tolerated by the specified application.

Standard Development Paradigms

A number of standard paradigms originating in the software community are available to use in modeling and simulation development. The most basic development paradigm, the waterfall paradigm, can be used to describe a straightforward new simulation development process consisting of the phases shown in the table below with their corresponding VV&A activities.

M&S Development Phases	V&V Activities
• Requirements definition	• Requirements verification
• Conceptual model development	• Conceptual model validation
• Design	• Design verification
• Implementation	• Implementation verification
• Integration and testing	• Results validation
• Release for use (Build)	• Accreditation

Although other paradigms are used to address more complex development situations, they are comprised of the same basic phases. Similarly, the V&V process associated with each paradigm will contain the same basic activities although the emphasis on each and the organization of the overall V&V effort will vary. In the case of iterative and more complex paradigms, interim "releases" are often made at the end of each iteration. Accreditation of these interim products is not normally formalized, although evidence of their fitness is generally needed for the development process to continue. Accreditation of the complete simulation is always tied to its fitness for use in a specific application.

In general, the greater the number of iterations (i.e., multiple passes through a development process or multiple repetitions of different phases of the development process) required by the development program, the greater the risk and uncertainty for the program and the greater the need for verification and validation.

The most common iterative paradigms are listed below. Additional information about each is available at the annexes listed below.

- incremental process paradigm
- prototyping paradigm
- evolutionary process paradigm
- spiral process paradigm
- re-engineering process paradigm

Paradigm Selection Factors

The development paradigm should meet the special needs of the M&S program. Some of the factors to consider in paradigm selection are listed in the table below:

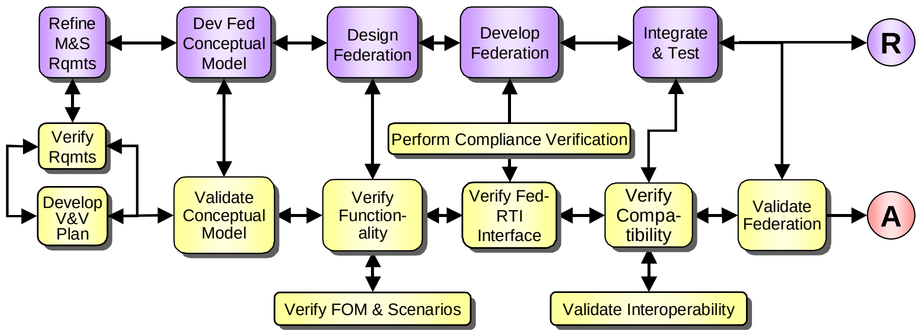
Paradigm Selection Factors
• organization of the overall program (one phase, multi-phase, etc.)
• complexity of the resulting simulation
• amount of time and resources available
• availability and allocation of resources, especially funding
• simulation category (legacy, new, federation)

Selecting an appropriate paradigm can greatly facilitate the entire development process. Selecting an inappropriate paradigm can result in increased difficulties with the development, including

- difficulties in meeting schedules and milestones
- ineffective testing and V&V activities
- inefficient allocation of resources
- increased program risk

Changing paradigms during the development process can also increase cost, cause delays, and confuse the V&V effort. All of which result in increased risk to the M&S program.

When the end product of an M&S development effort is a stand-alone model or simulation, there is more flexibility in selecting the development paradigm than when the end product will be part of a federation. A stand-alone model or simulation is seldom driven by outside constraints and considerations. Moreover, most federations consist of combinations of federates, each employing a different paradigm in their individual development or modification effort. The resulting federates become components of the overall federation through the federation development and execution process (FEDEP[1]), illustrated below, which serves as a federation development paradigm.



Federation Construction Paradigm

10/26/00

The development paradigm to use for a specific development program is the one that most closely captures the unique set of circumstances, constraints, schedule, resources, degree of persistence, and overall requirements and objectives of the program. In general,

- low risk programs involving very specific requirements and well-known legacy simulations can use a simple "once through" paradigm (e.g., waterfall paradigm)
- programs that involve evolving requirements, significant new M&S components, new applications, and/or complex configurations are likely to need a more complex paradigm (e.g., evolutionary)
- when interim products are involved, an incremental paradigm is most effective
- when time is limited, prototyping or even rapid prototyping may be the best solution

Paradigm Effects on the V&V Effort

The paradigm selected for the M&S development impacts how the V&V effort is planned and executed. Each V&V effort can be viewed as a process tailored by a number of factors such as those listed in the table below. Several of these are influenced by the paradigm used.

[1] One purpose for the FEDEP and its rigorous insistence on tools and standard approaches is to enable the federation developer to bring all the components together into a single overall paradigm.

Typical V&V Tailoring Factors
• type of program
• type of development effort*
• program schedule*
• resource availability*
• intended application
• amount of risk and uncertainty*
• intended persistence of the resulting simulation or federation
* influenced by the development paradigm

The development paradigm should be used during V&V planning to help determine how the V&V effort should be tailored -- what specific tasks should be done and in what order -- and to estimate what resources will be required. The V&V effort is driven by the development schedule. If development activities are delayed or must be redone because of problems with the paradigm, the entire process slows down and the V&V schedule, plans and activities must be adjusted to continue to meet the needs of the application. Federation integration, in particular, is a difficult, time-consuming activity and some delays and disruptions are to be expected. The V&V effort should anticipate such problems and develop a plan that is flexible enough to accommodate them.

Analysis of the development paradigm can provide information about the maturity of and uncertainties associated with the development effort and, therefore, the risks involved. The appropriateness of the development paradigm should also be considered as a factor when calculating risk and establishing priorities for both the V&V effort and the accreditation assessment. By adapting to the development paradigm, the V&V effort can concentrate resources on the most effective activities, tools, and procedures. Otherwise, inappropriate tasks are likely to be performed, tools and techniques misapplied, incorrect information gathered, and invalid conclusions drawn.

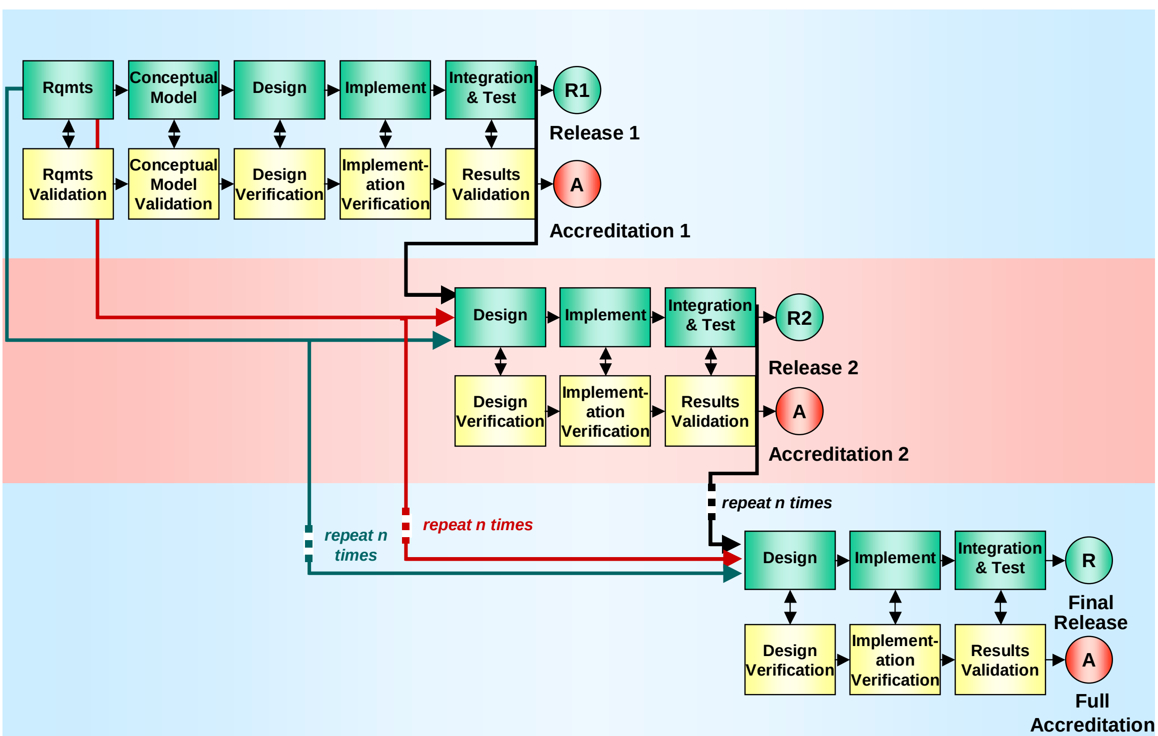
Paradigm Effects on Planning, Scheduling, and Costing

The development paradigm has a significant influence on planning and scheduling. Planning and cost estimating of iterative development programs – those that involve more than one pass through a development process or require multiple repetitions of different phases of the development process – should be handled differently than those that involve a single pass through the development process (i.e., waterfall paradigm). Iterative development programs vary considerably in the number of iterations involved: Some programs call for only two or three; others have been known to need as many as nine or ten iterations. Each subsequent iteration uses the information obtained from the previous iteration and adjusts the activities involved based upon this information.

Although general plans and cost estimates can be developed initially, they should not be relied on heavily during later development cycles. When possible, detailed planning and cost estimating should be done one cycle at a time. Planning, scheduling and cost estimation for each increment of an M&S development should account for the software and hardware inherited from previous iterations and should provide flexibility, time and resources to accommodate breakage and rework of problem areas.

An incremental development is normally more costly than a single cycle development effort. However, because an incremental development allows more time for testing and experimentation and more participation by the user community, weaknesses can be discovered early and corrected before they impact the final product. Similarly, the additional time and the controlled, graduated approach to the development can lead to more thorough and detailed validation and accreditation efforts and to a more credible simulation.

Example: VV&A in an Iterative Paradigm



Example Incremental Process

10/26/00

The example below illustrates the interaction between an M&S development process and V&V effort. It follows the basic pattern of an incremental development paradigm with three cycles or increments shown in the figure below.

Iteration 1

An interim product is needed to ensure the basic simulation will meet the needs of the User. Because Release 1 is not intended as an official release but as an alpha version, no accreditation assessment is considered necessary. Upon completion, iteration 1 is released to the user community for review and feedback is collected on the features and capabilities of the simulation. Recommendations and corrections are fed back to the appropriate phases for resolution.

Iteration 2

The second build does not have a requirements or conceptual model phase because these are inherited from the first build. Planning, scheduling and cost estimation for iteration 2 should take this reuse into account when scoping the effort to produce release 2. Because the requirements and conceptual model were verified and validated in iteration 1, the V&V effort for iteration 2 can focus primarily the remaining three development phases and should be planned and costed accordingly. The simulation is more fully developed during the second increment and the level of effort (LOE) depends on the complexity of the design and the amount of new code developed. Because iteration 2 is to be released as a beta version, an accreditation assessment is performed based on the requirements addressed by the first and second iterations. Problems identified during the V&V process or during beta testing are fed back to the appropriate phases for resolution. If the problem deals with issues inherited from increment 1, then its resolution may involve revisiting each of the phases in increment 2.

Increment 3

The third (and final) build essentially repeats the process involved in the second build. Because the more difficult design and code issues are generally delayed until the end, each phase should be carefully planned and scoped. The verification process involved would be similar to that defined for increment 2 but the validation and accreditation should cover more functionality, more complex interactions, and, in general, a greater scope.

References

Lewis, Robert O., *Cost Estimating Tutorial and Rationale Guide*, Tec-Masters, Inc., Huntsville, February 25, 2000. V3.0

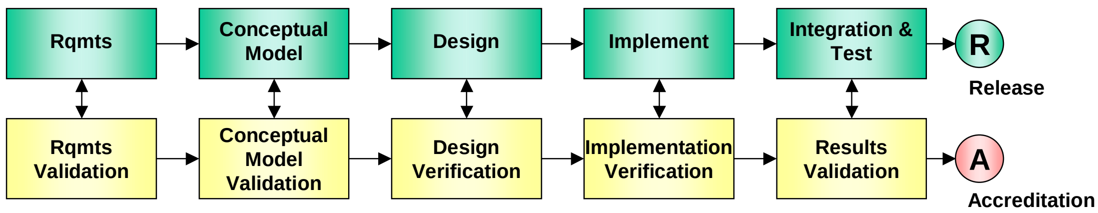
Tucker, William, "Chapter 4: Defining the Needs for Models and Simulations", *Applied Modeling & Simulation: An Integrated Approach to Development & Operation*. Cloud, D. and Rainey, L., editors, U.S. Air Force Academy, 1996.

RPG References in this Document

select menu*: RPG Special Topics*, select item: “Risk and Its Impact on VV&A”

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix A: Waterfall Paradigm



Waterfall or Grand (single pass) Paradigm

10/26/00

This figure is a representation of the basic waterfall paradigm. Feedback paths are not displayed and no allowance for overlapping phases is indicated. Overlaps between phases is an acceptable practice as long as unresolved problems and issues are not allowed to linger long enough to impact subsequent phases.

The waterfall paradigm is used when the requirements, conceptual model, and design are considered stable, predictable, and certain. It is appropriate when the simulation under development has predictable performance expectations, includes reused and previously accredited legacy components or federates, has pre-existing input data, experienced knowledgeable participants, and adequate documentation. Of all these factors, requirement stability and maturity are the most important.

The waterfall paradigm is the most common development paradigm for simulation except for very large, complex M&S efforts (e.g., multi-user or joint M&S programs). The waterfall paradigm is popular when modifying a legacy simulation or building a new model or simulation from an existing one because of the certainty of requirements, performance, and behaviors inherent in the legacy simulation.

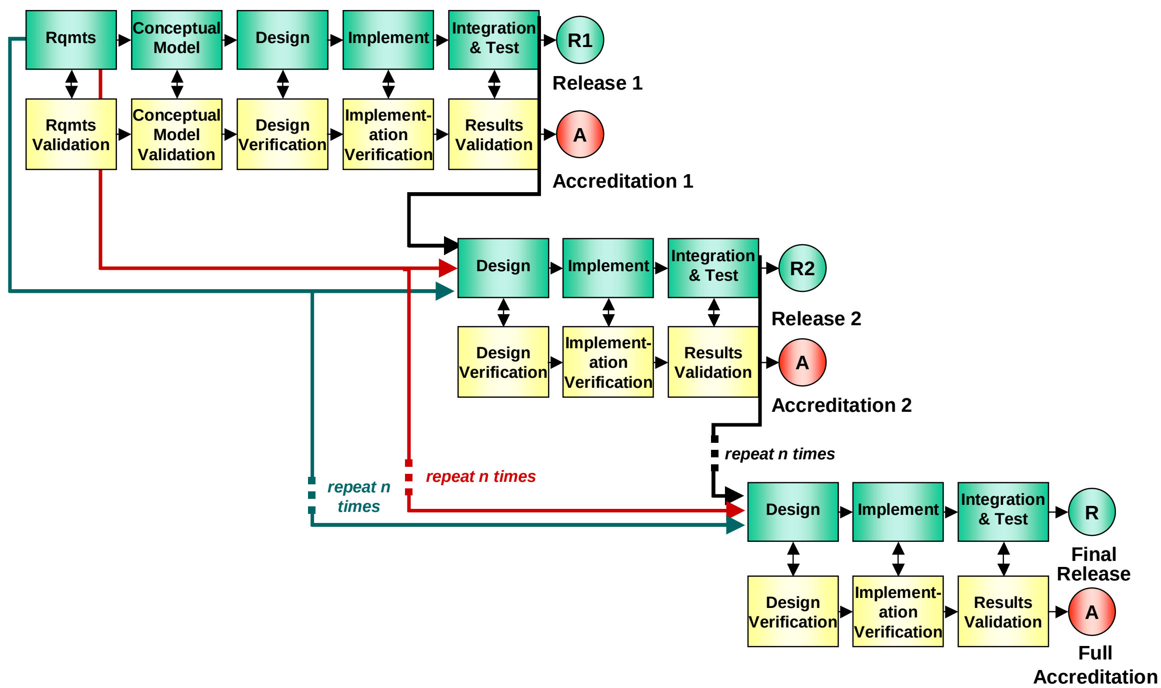
The V&V effort tends to be low to moderate, depending on

- amount of legacy code to be reused
- complexity of the simulation and fidelity required by the application
- level of risk and uncertainty associated with the application
- constraints on time and resources

The low range generally applies to the reuse of legacy simulations requiring little or no modification and the moderate range applies to more major legacy modifications and simple new developments. As long as the requirements are well understood, V&V costs can be quite economical.

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix B: Incremental Process Paradigm



Incremental Process Paradigm

10/26/00

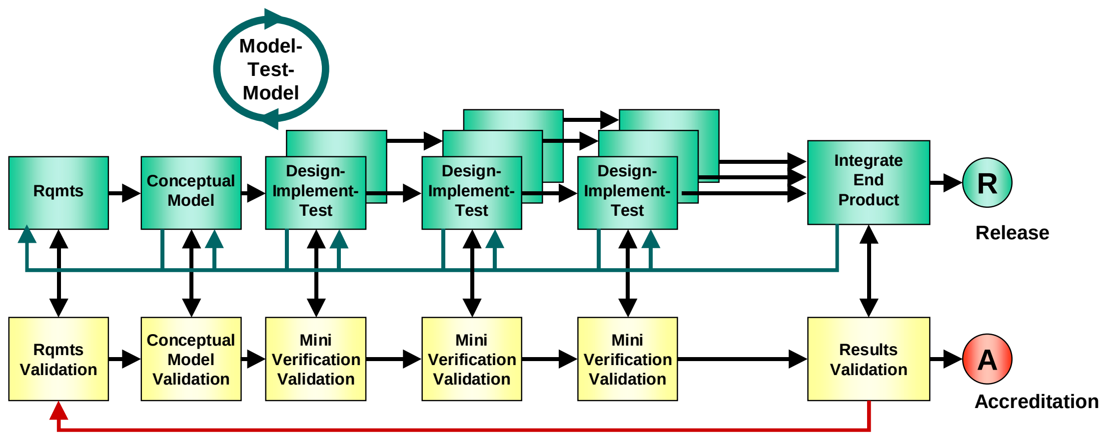
In an incremental process, not all increments, or builds, contain the same phases. In the example shown, there is only one requirements phase and conceptual model phase to support all three increments because of an assumption that minimal modification will be needed in either. The figure also shows no accreditation activity for the first increment, indicating that the first release is internal only.

The incremental paradigm is appropriate to use when a new model or simulation is being developed or when extensive reuse is employed, in particular when interim or partial builds (releases) are needed or considered desirable. Each increment or build involves a separate design-code-test (D-C-T) sequence. The requirements for each build should be relatively well defined, but those for the early builds are more likely to be stable. Verification and validation activities are conducted for each build; however, if an interim build is intended only for alpha testing (e.g., the release is limited to select members of the user group or no actual release is involved), an accreditation assessment may not be necessary.

In general, the V&V effort during an incremental development effort tends to be less vigorous for the early builds and moderately intensive for the final build. The costs of the V&V activities for each build will vary depending on what V&V activities are involved. In the example shown, the cost of the second and third build may be only 60% or 70% the cost of the first increment.

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix C: Prototyping Paradigm



Prototyping Paradigm

10/26/00

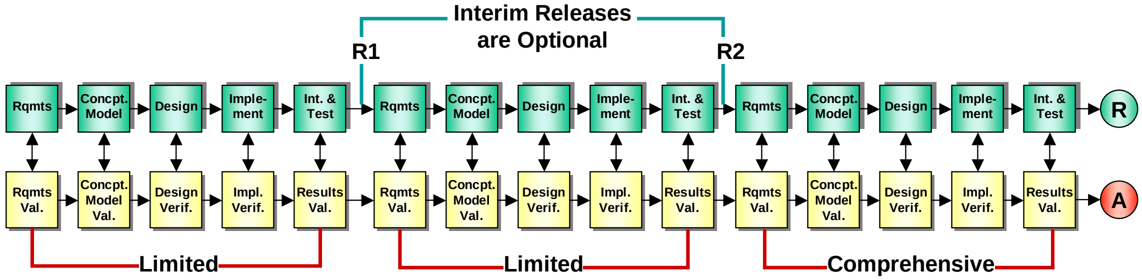
The prototyping, or rapid prototyping, paradigm is one of the most effective developmental approaches when requirements are not completely defined. The paradigm utilizes the design-code-test (D-C-T) sequence in each build. These provide an opportunity for a lot of interaction between the User, Developer, and V&V Agent. Rapid prototyping allows the preparation of quick, partial representations of the desired functionality that provide an indication of the intended performance, appearance, etc. for quick analysis and response. Prototyping is used when the requirements cannot be defined completely at the beginning of the program and the User has to participate in expanding and refining them as the development effort proceeds.

Prototyping is typically used in the development of new simulations to generate more specific requirements and specifications or to demonstrate alternatives; but it can also be used to demonstrate how a modification can perform in an existing simulation before committing to a complete rework effort. Although individual prototypes can be discarded at the end of their development phase, the end product does derive either directly from the prototypes themselves or from the lessons learned during their development

In either case, the V&V effort in a rapid prototype development focuses less on the actual implementation of the prototype and more on the lessons learned from working with the User and Developer regarding how to improve the requirements, conceptual model and design of the final product. Verification and validation activities tend to be minimal on the early prototypes, where the prevailing validation technique is face validation with User participation, and moderate on the final release.

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix D: Evolutionary Process Paradigm



Evolutionary Process Paradigm

10/26/00

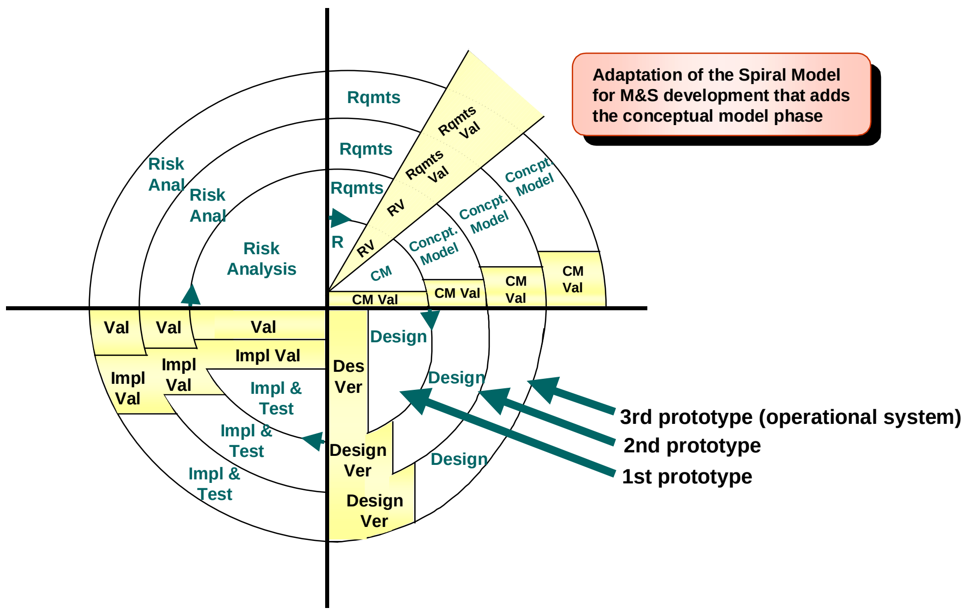
The evolutionary process paradigm is used when requirements cannot be completely defined initially and must evolve during development. It is particularly appropriate when part of the simulation must be built and tested in order to define the requirements for the rest. This paradigm allows the development process to iterate until the User and Developer are satisfied. This increases the User's confidence in the fitness of the simulation but makes it difficult to plan, schedule, and budget because of the uncertainty of the number of increments.

In this paradigm, the development process works across the entire breadth of the simulation before focusing on details (depth) [Tucker, 96]. Interim versions of the final product are generated and released for limited use. These are used to identify what additional requirements, features, functions, etc. are needed to achieve appropriate depth in the final product. Although this paradigm is often chosen because it helps reduce program risk, it introduces schedule and cost uncertainties because of the difficulty of estimating the impact of the additions on subsequent phases of the development.

Because it is assumed that detail and complexity grow uniformly across the simulation as it evolves and because accreditation is not generally needed for interim versions, the V&V effort for each interim evolution (not requiring accreditation) should be at a low level of intensity. However, more importance is placed on the final product (and any interim version requiring accreditation) and the intensity of the V&V effort increases to a moderate-to-high level of intensity accordingly.

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix E: Spiral Process Paradigm



Spiral Process Paradigm

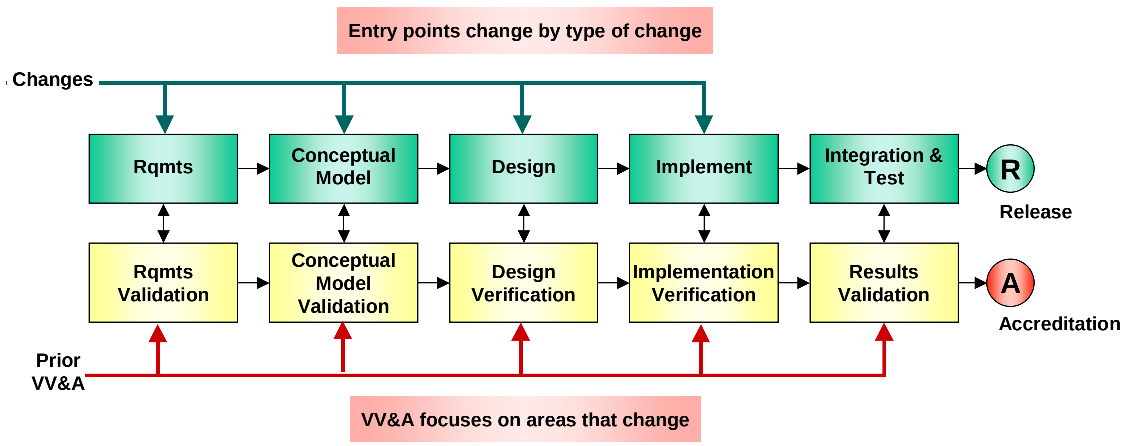
10/26/00

The spiral process paradigm is a version of the evolutionary paradigm that focuses on the depth in a few subsystems (components, models) at a time rather than on the overall breadth of the simulation. Some functions and capabilities are developed in the first iteration, others in the next, and so on until the simulation is completed. The spiral development paradigm has its own set of uncertainty problems and is difficult to cost, schedule, and evaluate because of the lack of cohesion in the simulation as a whole [Tucker, 96]. In addition, during the early stages of development, this paradigm tends to use simulation involving only simple or token representations. These may lack credibility in the eyes of Users resulting in less confidence in the fitness of the simulation being developed.

The spiral process paradigm presents a significant V&V challenge because of the high level of uncertainty in the direction and scope of the development. During the early stages of the spiral, the V&V effort is normally a low-level effort because of time constraints and the relative crudeness of the components being built. This increases to a moderate-to-high level of effort during the final iterations as more requirements are identified and the development effort becomes more focused.

In the web-based version of this document, the appendix below appears as a hot link in the Standard Development Paradigms section.

Appendix F: Re-engineering Process Paradigm



Re-Engineering Process Paradigm

10/26/00

The re-engineering process paradigm is used when legacy simulations are being modified for use in a different application. This paradigm is very similar to the waterfall paradigm except that it includes multiple entry points selected on the basis of what needs to be changed in the legacy simulation.

The overall V&V effort is minimal-to-moderate depending on the complexity and risks associated with each of the changes. Normally, little or no V&V is performed on the unchanging parts of the legacy simulation if the simulation and its VV&A history are well-documented and if there is a lot of similarity between the current and previous applications. However, if documentation is lacking or there are doubts regarding the fitness of the legacy simulation for the current application, then additional V&V activities may be required. The sections of the legacy simulation being modified should undergo a fairly moderate level of V&V and regression testing should be used to revalidate the unchanged parts of the simulation to ensure nothing has been corrupted. Once modification of the legacy simulation has been completed, the entire simulation should be validated based on current application.

§ § § § § § §